

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**A COMPARISON OF COMPUTATIONAL COGNITIVE
MODELS: AGENT-BASED SYSTEMS VERSUS RULE-
BASED ARCHITECTURES**

by

Craig L. Oeltjen

March 2003

Thesis Advisor:
Second Reader:

Rudolph Darken
Barry Peterson

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 2003	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: A Comparison of Computational Cognitive Models: Agent-Based Systems versus Rule-Based Architectures			5. FUNDING NUMBERS	
6. AUTHOR(S) Oeltjen, Craig L.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) Increased operational costs and reductions in force size are two of the major factors driving the need for improved computer simulations within the military community. Human performance models are used in various aspects of simulation, including controlling computer generated forces, tactical decision aides, intelligent tutoring systems and new system design. This research makes a comparison between two categories of human performance models, multi-agent systems and rule-based architectures. Each type of model has its own strengths and weaknesses, and is therefore better suited for certain applications. Complex military simulations need human performance models that take advantage of the strengths of more than one type of model. The purpose of this research is to compare the implementation and performance of these two models, and to demonstrate the need for hybrid systems that employ the best aspects of models for a given situation.				
14. SUBJECT TERMS Human Performance Modeling, Computational Cognitive Architectures, Human Factors, Decision Making, Multi-Agent Systems, Naturalistic Decision Making			15. NUMBER OF PAGES 80	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**A COMPARISON OF COMPUTATIONAL COGNITIVE MODELS:
AGENT-BASED SYSTEMS VERSUS RULE-BASED ARCHITECTURES**

Craig L. Oeltjen
Lieutenant Commander, United States Navy
B.S., North Dakota State University, 1988

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN MODELING,
VIRTUAL ENVIRONMENTS AND SIMULATION**

from the

**NAVAL POSTGRADUATE SCHOOL
March 2003**

Author: Craig L. Oeltjen

Approved by: Rudolph Darken
Thesis Advisor

Barry Peterson
Second Reader

Rudolph Darken
Chairman, Modeling, Virtual Environments and Simulation

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Increased operational costs and reductions in force size are two of the major factors driving the need for improved computer simulations within the military community. Human performance models are used in various aspects of simulation, including controlling computer generated forces, tactical decision aides, intelligent tutoring systems and new system design. This research makes a comparison between two categories of human performance models, multi-agent systems and rule-based architectures. Each type of model has its own strengths and weaknesses, and is therefore better suited for certain applications. Complex military simulations need human performance models that take advantage of the strengths of more than one type of model. The purpose of this research is to compare the implementation and performance of these two models, and to demonstrate the need for hybrid systems that employ the best aspects of models for a given situation.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	MOTIVATION	1
1.	Military Simulation Requirements.....	1
2.	Naturalistic Decision Making.....	1
3.	The Need for Hybrid Systems.....	2
B.	THESIS QUESTIONS.....	3
C.	APPROACH.....	3
D.	THESIS ORGANIZATION.....	4
II.	BACKGROUND AND RELATED WORK.....	5
A.	NATURALISTIC DECISION MAKING.....	5
1.	Naturalistic Decision Making Characteristics.....	5
2.	Applying NDM to Military Environments	6
3.	Levels of Expertise	6
B.	COMPUTATIONAL COGNITIVE ARCHITECTURES.....	8
1.	ACT-R.....	9
2.	Soar.....	10
3.	COGNET	11
4.	JESS	12
5.	OMAR.....	14
6.	Micro Saint	15
7.	Neural Networks	16
8.	Agent-Based Systems.....	17
C.	TOWARD HYBRID SYSTEMS	18
III.	DOMAIN SELECTION AND DESCRIPTION.....	21
A.	NDM APPROACH APPLIED TO SUBMARINE OPERATIONS	21
1.	Complex Domain.....	21
2.	Experienced and Knowledgeable Decision Makers	22
3.	Situation Assessment	23
B.	SUBMARINE OPERATING ENVIRONMENT	23
1.	Operating Area and Tasking	23
2.	Sensor Capabilities and Maneuvering Options.....	23
C.	OOD DECISION MODEL DESIGN CONSIDERATIONS.....	24
D.	IDENTICAL DOMAIN FRAMEWORK.....	24
1.	Operating Area and Horizontal Coordinate System	24
2.	Bottom Contour and Fathometer Capability	24
3.	Obstacles and Sonar Capabilities.....	24
4.	Starting Position and Destination Point.....	25
5.	Course, Speed and Depth Change Options.....	25
6.	Mission Tasking and Operating Goals.....	25

IV.	AGENT-BASED OOD DECISION MODEL IMPLEMENTATION	27
A.	DESCRIPTION AND DESIGN.....	27
B.	MANEUVERING ACTION SETS.....	27
	1. Critical Action Sets	27
	2. Moderate Action Sets.....	29
	3. Active Action Sets	29
C.	DECISION PROCESS	30
	1. Setting Operating Goal Weights.....	30
	a. <i>Avoid Grounding Goal</i>	30
	b. <i>Avoid Collision Goal</i>	31
	c. <i>Remain Within Operating Area Goal</i>	31
	d. <i>Transit to Destination Goal</i>	31
	2. Active Action Set Selection and Conflict Resolution	31
	3. Maneuvering Commands	32
	a. <i>Avoid Grounding Goal</i>	32
	b. <i>Avoid Collision Goal</i>	32
	c. <i>Remain Within Operating Area Goal</i>	32
	d. <i>Transit to Destination Goal</i>	32
D.	LEARNING PROCESS.....	33
	1. Action Set Performance Evaluation	33
	a. <i>Avoid Grounding Goal</i>	33
	b. <i>Avoid Collision Goal</i>	33
	c. <i>Remain Within Operating Area Goal</i>	33
	d. <i>Transit to Destination Goal</i>	34
	2. Assigning New Active Action Sets	34
V.	RULE-BASED OOD DECISION MODEL IMPLEMENTATION.....	35
A.	DESCRIPTION AND DESIGN.....	35
B.	KNOWLEDGE BASE FACTS.....	35
	1. Fact Templates	35
	a. <i>Status Fact Template</i>	36
	b. <i>Destination Fact Template</i>	36
	c. <i>Obstacle Fact Template</i>	36
	d. <i>Shoal Fact Template</i>	36
	e. <i>OutArea Fact Template</i>	36
	2. Asserting Facts	37
C.	KNOWLEDGE BASE RULES.....	37
	1. Avoid Grounding Goal Rules.....	38
	2. Avoid Collision Goal Rules	38
	3. Remain Within Operating Area Goal Rules	39
	4. Transit to Destination Goal Rules	39
D.	OPERATING GOALS AND CONFLICT RESOLUTION.....	40
	1. Scenario Tracking Facts.....	40
	2. Scenario Rule Saliency.....	40
E.	EXPLAINING OODDM ACTIONS	41
	1. The JESS Watch Function	41

2.	OODDM Application of the Watch Function	41
VI.	OOD DECISION MODEL COMPARISON.....	43
A.	COMPARISON PLAN.....	43
B.	GRAPHICAL INTERFACE.....	43
1.	Display Window	43
2.	Operating Data Dialog Box.....	43
3.	Control Buttons.....	44
4.	DOS Display Window	45
C.	IMPLEMENTATION CONSIDERATIONS.....	45
1.	Agent-Based OODDM.....	45
2.	Rule-Based OODDM	46
D.	PERFORMANCE AND OBSERVED BEHAVIOR	46
E.	COGNITIVE FUNCTION REPRESENTATION.....	47
1.	Sensing and Perception.....	47
2.	Long and Short-Term Memory	47
3.	Multi-Tasking Ability and Decision Making.....	48
4.	Learning (Agent-Based OODDM Only)	48
5.	Ability to Explain Actions (Rule-Based OODDM Only).....	49
F.	COGNITIVE VALIDITY	49
VII.	CONCLUSIONS AND RECOMMENDATIONS.....	51
A.	COMPARISON CONCLUSIONS	51
1.	Environment Complexity Can Overwhelm.....	51
2.	Learning is Powerful.....	52
3.	Explanations Must be Useful	52
B.	RECOMMENDATIONS FOR FUTURE WORK.....	53
1.	More Complex Environment or Goals.....	53
2.	Improved Agent-Based OODDM	53
3.	Improved Rule-Based OODDM	53
APPENDIX A.	AGENT-BASED OODDM PSEUDOCODE	55
1.	SENSING SYSTEM INPUTS.....	55
2.	SETTING OPERATING GOAL WEIGHTS	55
3.	ACTION SET SELECTION AND CONFLICT RESOLUTION	56
4.	MANEUVERING COMMANDS AND MOVEMENT	57
	LIST OF REFERENCES.....	59
	BIBLIOGRAPHY.....	61
	INITIAL DISTRIBUTION LIST	63

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	JESS Fact Templates.....	36
Figure 2.	JESS Watch Function Reports	42
Figure 3.	Operating Area Display Window.....	44

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Constant Domain Parameters.....	26
Table 2.	Critical Action Sets	28
Table 3.	Moderate Action Sets.....	29
Table 4.	Initial Active Action Sets.....	30
Table 5.	Maneuvering Parameter Changes For Operating Goal Scenarios	38

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS

ACT-R	Adaptive Control of Thought – Rational
AETS	Advanced Embedded Training System
COGNET	Cognition as a Network of Tasks
D-COG	Distributed Cognition
DTED	Digital Terrain Elevation Data
EOOW	Engineering Officer of the Watch
ITS	Intelligent Tutoring System
JESS	Java™ Expert System Shell
MA&D	Micro Analysis and Design
MAS	Multi-Agent System
NDM	Naturalistic Decision Making
NIMA	National Imagery and Mapping Agency
OMAR	Operator Model Architecture
OOD	Officer of the Deck
OODDM	Officer of the Deck Decision Model

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

1. Military Simulation Requirements

Reductions in military operating budgets and improvements in computer technology have resulted in the increased use of simulations throughout the military. Simulations are being employed in a wide variety of applications including training, mission rehearsal, system analysis, system acquisition and tactical decision aiding. The human behavior representation or human performance model is a key component in each of these simulations. In some cases, such as controlling computer generated opposition forces, the model is only required to generate behavior that meets the users' expectations. There is no need for an explanation of how or why the behavior was generated. In other cases, such as intelligent tutoring systems (ITS), the model should be able to provide a detailed rationale for each of its actions. This rationale is used to provide feedback to the student and may include perceived sensory inputs, task priorities, previous experiences, expected consequences of the action, individual personality characteristics or any other pertinent information.

There are two other major trends within the military that are also contributing to the need for improved human performance models. First, information technology and information warfare must be considered in simulations. This implies a need for a model that perceives, interprets and responds to information in the same way as a real human, because humans are the focus of information utilization. Second, the use of distributed simulations presents a need for models that act as collateral friendly forces when there are not enough individuals available to fill all of the positions.

2. Naturalistic Decision Making

Developing human behavioral models that can meet the needs of the military simulation community is a very difficult problem. The computational models must be able to make decisions in complex, dynamic domains and be able to explain their actions,

like real humans. As computational models begin to more closely match the actual human cognitive processes, the descriptive model on which they are based becomes more critical. Traditionally, computational models have been based on a descriptive model called Rational Choice Theory (Zsombok, 1997). The decision maker generates a list of several possible actions, and then chooses the best action from the list. The transition from the descriptive rational choice theory model to a computational model is relatively straightforward. The problem space is decomposed into various states and transition links. A search algorithm is then used to find possible paths from the current state to a given goal state.

While rational choice theory may be applicable in many environments, recent studies of experienced decision makers in more complex situations suggest a theory called Naturalistic Decision Making (NDM). Rather than focusing on the possible options, the decision maker focuses on thoroughly assessing the situation and then draws on his knowledge and experience of the problem environment to make his decision. Making the transition from the descriptive NDM theory to a computational model raises several questions: Who is an experienced decision maker? How do you model the progression from being a novice to an experienced decision maker? How does the model account for previous experiences? Are there situations where rational choice theory should still be used? These questions and many others remain unanswered, and an NDM based computational model has not yet been developed. But, given the direct applicability of NDM to military command and control decision environments, there is a clear requirement for this type of model.

3. The Need for Hybrid Systems

Every problem domain has associated with it certain aspects of human cognition that are critical to task performance in that environment. These cognitive aspects may include attention, memory, learning, multi-tasking, planning, situation assessment, or the underlying descriptive theory of decision making. As a model designer, the first step is to select a cognitive architecture that can accurately represent the cognitive aspects that he has determined to be critical for his application. As environments become more

complex, more of the cognitive aspects must be accurately represented, so an acceptable model must provide cognitive validity over a wider range of capabilities. Chapter II contains detailed reviews of several cognitive architectures. These reviews show that each architecture has its own strengths and weaknesses, but that none of them can adequately represent all aspects of human cognition that translates into accurate, representative behavior. A logical near-term approach to this problem might be to combine the strengths of two or more architectures to produce a hybrid that better represents overall human cognition without sacrificing behavioral fidelity.

B. THESIS QUESTIONS

This thesis will address the following questions:

- What portions of human behavior are best represented using rule-based computational cognitive architectures?
- What portions of human behavior are better suited for representation using agent-based computational cognitive systems?

Linking the various types of computational architectures with the aspects of human behavior for which the architecture is best suited is the first step in the development of more robust computational cognitive models. These relationships between cognitive requirements and architecture types will lay the foundation for the development of hybrid systems that can incorporate and integrate the best qualities of several architectures.

C. APPROACH

The purpose of this thesis is to continue researching the development of hybrid computational cognitive architectures by investigating the difference in performance of two models designed to operate in the same problem space. The models are required to select between several shifting goals and choose from several possible actions to accomplish each goal. The first model uses an agent-based design. The second model

employs a popular rule-based shell. The problem space is based on submerged submarine navigation and includes operating area boundaries, navigation hazards (obstacles) and a contoured ocean floor. The models are allowed to change the submarine's course, speed and depth to safely execute the given mission. The models' performance will be compared and evaluated, investigating the differences between the models and addressing the strengths and weaknesses of each individual model.

D. THESIS ORGANIZATION

This thesis is organized as follows: Chapter II discusses the background of Naturalistic Decision Making and describes several cognitive architectures and computational methods currently in use. Chapter III describes the submarine navigation domain chosen for this research. Chapter IV discusses the implementation of the agent-based model. Chapter V discusses the implementation of the rule-based model. Chapter VI compares the two models. Chapter VII provides conclusions and recommendations for future work.

II. BACKGROUND AND RELATED WORK

A. NATURALISTIC DECISION MAKING

1. Naturalistic Decision Making Characteristics

When considering descriptive models of human decision making, the one that is most widely accepted and commonly used is the rational choice theory. It states that people generate several possible courses of action, make comparisons between them, and then select the best alternative (Zsombok, 1997). Computational models based on rational choice theory typically use a search algorithm to generate all possible solutions to a given problem. Each solution is assigned a score that is based on domain specific characteristics, such as cost or time to completion. The best solution is then chosen by comparing these scores. Research of decision making in complex environments shows that experienced people do not follow the rational choice theory. They make decisions based on a thorough assessment of the current situation and past experiences they have had that are similar. Gary Klein and Caroline Zsombok explain this behavior using the Naturalistic Decision Making (NDM) approach. NDM focuses on how people use their past experience and domain knowledge to quickly make decisions in complex situations.

Klein and Zsombok use four key aspects to define situations to which the NDM approach can be applied. First, the complex domain is characterized by ill-structured problems, a dynamic environment, competing goals, time stress and high stakes. Second, the decision makers are experienced in and very knowledgeable about the problem domain. Third, the actual decision may not be the most important issue. The situation assessment and its relationship to past experiences also provide key information. Fourth, the purpose is to explain how experienced people make decisions, not to provide a method that can be used by less experienced people to make these same decisions (Zsombok, 1997).

2. Applying NDM to Military Environments

As defined above NDM can be applied to a broad spectrum of military decision making environments. At one end of the complexity spectrum consider a sonar watch supervisor onboard a navy vessel. He is the leader of a small watchteam. He receives inputs from two or three subordinates, makes decisions on the employment of their resources, and makes reports only to his direct supervisor. At the other extreme is the joint task force command center watch captain. His inputs can come from numerous sources that range from the 30 to 40 subordinate watchstanders in the command center to the many outside agencies working for him throughout the theater of operations. Each of these examples contains the key characteristics of an NDM domain: ill-structured problems, a dynamic environment, competing goals, time stress and high stakes.

3. Levels of Expertise

If the NDM model is only appropriate for experienced decision makers (Zsombok, 1997), some method must be used to describe and identify an expert. Hubert and Stuart Dreyfus have developed a five-stage model of skill acquisition to describe the differences between individuals with different levels of competence and experience (Dreyfus, 1997). They use two environments, car driving and chess playing, to illustrate the progression of both motor skills and intellectual skills.

In the first stage, novice, the instructor reduces the task environment to its simplest elements so that the beginner can recognize and understand them without any previous experience in the task domain. A rule set is then provided to determine actions based on the state of these simple elements, similar to a computer executing a program. For example, a novice driver may learn to shift into second gear when the speedometer indicates ten miles per hour. Novice chess players learn numerical values for each piece and a general rule to always exchange if the total value of the pieces captured is greater than the total value of the pieces lost.

After seeing a number of examples and gaining some experience in real situations the novice progresses to the second stage, advanced beginner. By now he has learned to

recognize relevant cues on his own and uses them, in conjunction with the novice rule set, to make better decisions. Advanced beginner drivers learn to shift gears based on the sound of the engine, rather than relying on the speedometer. Advanced beginner chess players learn to recognize less desirable positions and how to avoid them.

With more experience the advanced beginner becomes overwhelmed with the number of potentially relevant cues he has learned to recognize. To reach stage three, competence, he must devise a plan to determine which cues are important and which can be ignored, allowing him to create a hierarchical perspective of the situation. Competent performers develop new rules to decide on a plan or perspective. But, given the vast number of possible situations he may encounter, each differing in subtle ways, it is not feasible to develop a rule list of what to do in every case. Because of this, competent performers must choose a plan without being certain that it will be successful. Task performance now becomes nerve-racking as the competent feels a great sense of responsibility for his actions. For example, a competent driver entering a curve may consider speed, surface conditions, space available and time constraints before deciding that the car is going too fast. Now he must choose to either let up on the accelerator or apply the brakes, and will be happy to get through the curve without going into a skid. A competent chess player may study the board and decide that her opponent's king is weakly defended and vulnerable to an attack. But, after choosing to attack she ignores weaknesses in her own position created by her maneuvers.

Achieving proficiency, stage four, requires the performer to incorporate his experience into his theory of the skill, replacing rules and principles with situational discriminations and associated responses. His behavior shifts from reasoning to intuition. Task performance is easier and less stressful now because he can simply see what needs to be accomplished, without having to evaluate several relevant cues. A proficient driver entering a curve knows intuitively when he is going too fast. She must still decide what action to take to slow the car, but valuable time has been saved because she did not have to specifically decide, based on several factors, that she was going too fast. Proficient chess players can recognize almost immediately and without conscious effort the strategic sense of a given situation, for example that attacking is the goal. But, they must still deliberate about how to best conduct the attack.

A performer who reaches the highest level, expertise, can not only intuitively see what goal needs to be accomplished, but can also intuitively see what actions should be taken to attain the goal. Enough experience in a wide variety of situations allows her to form classes of situations that share the same decision, action or tactic. This classification allows the immediate intuitive response that is characteristic of an expert. When faced with a novel situation an expert may revert to level of competence or proficiency because he has not experienced enough similar situations to establish this intuition. Expert drivers not only know that the car is going too fast, but also take the appropriate action on the accelerator and brake pedals with little, if any, conscious effort. Expert chess players can recognize up to 50,000 types of board positions and can play at a rate of less than ten seconds per move without any serious degradation in performance (Dreyfus, 1997).

B. COMPUTATIONAL COGNITIVE ARCHITECTURES

This section provides overviews of several popular computational cognitive architectures by comparing the following key design aspects: purpose and use, theoretical assumptions, architecture and functionality, operation, current implementation and support environment. The material draws heavily on the work of the Panel on Modeling Human Behavior and Command Decision Making: Representations for Military Simulations. The panel was established by the National Research Council in 1996 to review the state of the art in human behavior representation as applied to military simulations, with emphasis on the areas of cognitive, team, and organizational behavior. The panel published an interim report (Pew & Mavor, 1997) and a final report (Pew & Mavor, 1998) that are both outstanding references for anyone working in this field. Chapter three of the panel's final report provides more detailed reviews of the architectures considered in this thesis and of several additional architectures not discussed in this work.

1. ACT-R

Adaptive Control of Thought-Rational (ACT-R) is one of the few architectures that was originally designed to be a model of higher-level cognition. It has been used in several domains including mathematical problem solving, maze navigation, and human memory and learning. ACT-R assumes that there are two types of knowledge, declarative and procedural, that exist permanently in long-term memory. Declarative knowledge is represented by structures with attributes. Production rules form the procedural knowledge. Working memory is the portion of declarative knowledge that is currently active, so it is limited by existence or access, not by capacity. When production rules are proposed in response to goals they retrieve information from declarative memory. While several production rules may be proposed, only one can take action on a given cycle. The conflict resolution system chooses the rule that will most likely lead to the best result. ACT-R also includes several learning mechanisms that can be turned on or off depending on the needs of the model. Declarative knowledge can be learned from the outside world or through problem solving. Associations between declarative memory elements can be tuned through experience. New production rules can be generated through analogy. Production rule strengths can be modified through experience. To build a model all initial declarative and procedural knowledge must be hand-coded, along with numerical values for production strength, production cost, probability of success of a given goal, and other parameters. The output of the model is a trace of productions that fired and the details of the declarative knowledge used by each production. If learning is turned on, the output also includes any newly created elements or modified parameters.

ACT-R has a large user group and an electronic mailing list to announce software improvements and related issues. ACT-R software, documentation, tutorials, and a number of implementation tools are available for downloading at the ACT-R homepage (<http://act.psy.cmu.edu>) maintained by Carnegie Mellon University (CMU). CMU also hosts an annual ACT-R workshop to allow researchers to present their work and discuss future developments. ACT-R is written in Lisp. In an effort to improve compatibility and portability an open source Java™ version, known as *jACT-R*, is under development.

Documentation and the beta release software is available at the *j*ACT-R homepage (<http://jactr.sourceforge.net>).

2. Soar

Soar is a symbolic cognitive architecture that implements goal-oriented behavior by searching a problem space, and learns from the results of its problem solving. Soar, like ACT-R, was originally developed as a unified theory of cognition for the purpose of modeling human problem solving and learning. Soar assumes that human behavior can be modeled as a cognitive processor that operates in conjunction with a perceptual input processor and motor output processor. Part of the Soar design philosophy is to minimize the number of distinct architectural mechanisms of the system. This results in a single mechanism for permanent knowledge known as productions, so there is no distinction between procedural and declarative knowledge. Working memory elements, objects with attributes and values, represent all temporary knowledge. The only learning method is chunking.

The basic Soar execution cycle consists of proposal, decision and application phases and, if necessary, input and output phases that interact with an external environment. The proposal phase interprets working memory element data to determine the current situation. Active working memory elements are used to determine if the initial conditions are met for production rules. The associated operators are proposed for all production rules whose initial conditions are met. The decision phase then weighs the preferences associated with each proposed operator to choose a new operator. Whenever the active working memory elements are not sufficient to allow a unique operator choice a new subtask is generated. The goal of the subtask is to resolve the impasse by either searching the problem space in an attempt to locate missing information or by making the decision based on the incomplete information. If the missing information is found, Soar's learning mechanism establishes a new association between that working memory element and the original task. Chunking in this manner transfers knowledge from the subtask space to the original task space, preventing the need for the execution of the subtask again in the future and allowing the model to inductively acquire new knowledge.

Soar models have been used extensively in large-scale military simulations. Specifically, in simulated theater of war (STOW-E in 1995 and STOW-97) Soar intelligent forces were used to simulate the behavior of pilots on combat and reconnaissance missions. Additional application areas have included natural language interpretation, visual search, and storage of key situations and decisions for use in post-task debriefing. Current research focuses on integrating Soar's agent architecture with complex, modern computer games to develop smart adversaries. Using a dynamically loaded library, the popular game Quake can interact with one or more Soar agents (Quakebots). In this project, the goal has been to determine what types of knowledge are necessary to build interesting opponents that make use of the same tactics and have the same abilities as their human counterparts.

Soar has a very large user group and a thorough support environment that includes user and reference manuals, a tutorial of models demonstrating various Soar capabilities, and editing and debugging tools. User interaction with Soar software is through a Tcl/Tk-based graphical user interface with panels for creating, monitoring and controlling agents. All documentation and software is available for downloading at the Soar homepage (<http://bigfoot.eecs.umich.edu/~soar/main.html>) maintained by the University of Michigan. The site also provides a list of frequently asked questions and points of contact for Soar projects.

3. COGNET

COGnition as a NEtwork of Tasks (COGNET) was originally designed to build user models for intelligent interfaces, focusing on cognitive rather than psychomotor tasks. COGNET assumes that humans perform multiple tasks in parallel, and these tasks compete for the human's attention. The appearance of parallelism is achieved through serial processing with very rapid attention switching; so several tasks may be in various stages of completion. The COGNET architecture consists of a problem context, a perception process, tasks, a trigger evaluation process, an attention focus manager, a task execution process, and an action effector. There is no explicit environment representation, rather COGNET interfaces with the external world through its shell. The

problem context is a multipanel blackboard system that serves as a means of communication between tasks. The perception process recognizes events in the external environment and posts relative information to the appropriate panel of the blackboard. Each independent task has a set of trigger conditions. When its conditions are satisfied the task is activated and becomes eligible to execute. It competes for attention based on the priority of the associated goal, which can depend on specific blackboard content. The action taken by a task can send data to or receive data from a blackboard or other device, or it can suspend the task until a specified condition exists and turn control over to other tasks while it waits. The attention focus manager monitors task priorities and shifts attention by controlling task state. It uses the task execution process to start, interrupt, and resume tasks, always ensuring that only the highest priority task runs at any time. The action effector makes changes to the external environment.

COGNET is a commercial product marketed by CHI Systems, Incorporated. An extensive support environment, GINA, is available that includes editing, debugging and testing tools. COGNET does not have the large user community seen with open source systems like ACT-R or Soar. The Advanced Embedded Training System (AETS), uses a COGNET model to implement an intelligent tutoring system for a ship-based combat information center, has been very successful in demonstrating COGNET's performance in a military command and control type environment. Two shortcomings must be considered, especially when developing models that require a high degree of cognitive fidelity. First, there is limited psychological validity supporting the "parallelism through rapid attention switching" theory. Second, COGNET does not allow for any learning by the model.

4. JESS

Java™ Expert System Shell (JESS) is a rule engine and scripting environment, written entirely in Sun's Java™ computer language that was developed by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, California. JESS was originally designed as a Java™ clone of NASA's CLIPS expert system shell, but it has evolved into a distinct, Java™ influenced environment of its own.

Rule-based programming is one of the most commonly used techniques for developing expert systems. In this programming paradigm, rules are used to specify a set of actions to be performed for a given situation. A rule is composed of an *if* portion and a *then* portion. The *if* portion of a rule is a series of patterns which specify the facts (or data) which cause the rule to be applicable. The process of matching facts to patterns is called pattern matching. The expert system tool provides a mechanism, called the inference engine, which automatically matches facts against patterns and determines which rules are applicable. The *if* portion of a rule can actually be thought of as the *whenever* portion of a rule since pattern matching always occurs whenever changes are made to facts. The *then* portion of a rule is the set of actions to be executed when the rule is applicable. The actions of applicable rules are executed when the inference engine is instructed to begin execution. This process continues until no applicable rules remain. Rule-based expert systems are extremely powerful because actions themselves can assert new facts. When this happens additional rules apply and their actions are executed.

The JESS knowledge base contains both facts and rules. A fact is a construct used to represent a piece of information that is known to be true. A rule is an *if/then* statement that defines the set of facts that must be true (the *if* part) before a set of actions (the *then* part) can be executed.

The JESS inference engine is based on a very efficient pattern matching method called the Rete algorithm. A simple inference engine implementation would keep a list of rules and continuously cycle through the list. For each rule it would compare the entire fact list to the *if* part, and execute the actions of the *then* part for those rules that are satisfied. This is inefficient because most of the tests made on each cycle will have the same results as on the previous iteration. However, since the knowledge base is stable, most of the tests will be repeated. The Rete algorithm reduces this inefficiency by remembering past comparison results across iterations of the rule loop. Only new facts are compared to the *if* part of rules. Additionally, by checking to see if the new facts create any groups of facts that are required to satisfy a rule, the new facts are only compared with the *if* parts of rules to which they are most likely to be relevant (Forgy, 1982).

Because JESS is a model development “shell” and production system only, it is not based on a specific cognitive architecture, and is most often used for purely declarative knowledge in expert systems. If the application has a requirement for a cognitive basis, the user is responsible for incorporating any behavioral or psychological validity.

JESS has a large user group and continues to grow in popularity. Sandia National Laboratories maintains a very informative website (<http://herzberg.ca.sandia.gov/jess/>) containing documentation, trial software available for downloading, lists of frequently asked questions, and points of contact.

5. OMAR

The Operator Model ARchitecture (OMAR) models human operators of complex systems for the purpose of procedural evaluation and system design analysis. In particular, OMAR was developed to model situated-cognition, where a human dynamically shifts between goals based on events occurring in the environment. OMAR is based on the following assumptions. Human behavior is goal directed and multiple tasks may be performed concurrently. But, because these concurrent tasks compete for limited sensory, cognitive and motor resources, parallel behavior is limited to the case in which some of the concurrent tasks are over learned to the point of automaticity. OMAR also assumes that operators work in teams and has provisions for modeling several communicating operators.

OMAR can be described as a set of interacting layers. The base is the core simulation layer which is a discrete-event simulator using time-sorted queues of future events. The perceptor/effector layer allows interaction with the environment. Default models of sensory and motor skills have been provided, but much more detailed models can be used when required. The cognitive layer consists of agent entities, each capable of executing their own goals, plans and tasks. Goals are decomposed into trees of subgoals. A plan is a set of leaves of a goal tree, and a task is the instantiation of a plan. Task priorities are computed on the basis of existing conditions. The task with the highest priority level executes until it is completed or is preempted by another task of higher

priority. Execution is generally serial, with the exception of some highly automated tasks that can run in parallel with others.

OMAR is a commercial product developed by BBN Systems and Technologies. A powerful toolkit provides a concept editor, a procedure browser and output analysis tools. While OMAR offers great flexibility and detail in behavior, it is very complex and model development is likely to be labor intensive. Although OMAR has not been used extensively in military simulations, its use in the future could increase with the continued development of the Distributed Cognition (D-COG) project by the Air Force Research Laboratories (Eggleston & Young, 2000). D-COG is based on a descriptive framework called cognitive system engineering that tries to preserve the ecology of a task by not reducing it to a single well-formed process. Using OMAR as the foundation, D-COG focuses on recognition and shaping, rather than procedures and processing. D-COG is still in the initial development phase and not nearly as mature as the others architectures discussed here, but it is an interesting new approach to the human performance modeling problem.

6. Micro Saint

Micro Saint is a discrete-event network simulation language used for the analysis of complex human-machine systems. Micro Saint is not a model of human behavior, but it is a simulation system with tools that can be used to create human behavior models to meet user needs. The outputs of these models are estimates of task completion times, task accuracies, and human operator workload. At the heart of the Micro Saint system is the network of tasks. The nodes of the network are the tasks. Micro Saint tasks are categorized as follows: visual, numerical, cognitive, fine and gross motor, and communications. For each task the user must provide a set of characteristics that includes probability distributions for accuracy and completion time, conditions that must be met to begin the task, changes in the system when the task begins and ends, and what action to take upon completion. The arcs of the network are task sequence relationships. The accuracy and completion times of each task are modeled stochastically using

parameters selected by the user. The workload associated with each individual task is aggregated to compute a composite workload measure.

Micro Saint is a commercial product marketed by Micro Analysis and Design, Incorporated (MA&D). Support software includes editors for constructing task networks, task descriptions and task completion decision logic. Because Micro Saint is a model development tool, the user is responsible for building a model that meets the behavioral and psychological validity requirements of the intended application. Micro Saint has been used extensively in constructive simulations in the analysis and design phase of military systems and for generating human performance tables that could be used in virtual simulations. MA&D recently started a new project that plans to incorporate Klein's NDM theory into the existing Micro Saint task network (Archer, Warwick and Oster, 2000).

7. Neural Networks

The general cognitive systems called neural networks, connectionist networks or parallel distributed processing systems are quite different from the other architectures considered in this section because they are more of a computational approach than a cognitive or behavioral architecture. Neural networks have been used to model a broad range of cognitive processes including pattern recognition, self-organization of stimuli, dynamic system control and solving prediction problems. Neural networks are motivated by principles of neuroscience and are based on the following two assumptions. Human behavior can be represented by self-organizing networks of primitive neural units and all complex human behaviors of interest can be learned by neural networks through appropriate training.

A neural network consists of several layers of abstract neural units, or nodes, that begin with an input layer that receives external stimuli and ends with an output layer that provides the system response. Each of the layers between the input and output layers contains a large number of nonlinear nodes to perform essential calculations. Connections can be made between nodes within a layer or between nodes in separate layers. When a stimulus is received, activation originates from the input layer, cycles

through the intermediate layers until they reach equilibrium, and produces activations at the output layer. Each connection has an associated connection weight. These connection weights are trained with extensive data sets that contain desired results for a wide range of input conditions. Learning algorithms are used to adjust the weights to maximize performance. The system is then tested on a data set not used in the training to evaluate its performance. The training process iterates until a satisfactory level of performance is attained. Neural networks use a distributed representation of information and knowledge. The activity pattern across the network nodes at any particular time represents the state of the dynamic system at that time. The state evolves until it reaches equilibrium. This final state represents all information retrieved from memory for the given input stimulus and the output layer activation represents the short-term memory. Long-term memory is represented by the connection weights between the nodes.

Neural networks have several strengths that are typically not associated with more traditional rule-based systems. Supervised learning is used to train connection weights, rather than having to modify or create production rules. Networks have been designed with real-time weight adjustments, allowing the system to respond to a non-stationary environment. The network's extensive distribution of information and parallelism provide robustness to noisy or totally new input stimuli. They also provide a high level of fault tolerance because an error at a single node or connection will probably only cause a slight degradation in overall performance (Haykin, 1994).

Most neural network applications have been focused on a small portion of the cognitive system, usually related to sensory and motor processes. Extending a neural network to model performance of a task with high-level reasoning involving structured domain information presents difficulties with encoding the large knowledge base and with the extensive training required to tune the connection weights.

8. Agent-Based Systems

Like neural networks, agent-based systems must be considered as more of a computational approach than a cognitive architecture. A software agent, as defined by Ferber has four important capabilities. First, through actions or communication, an agent

must be capable of interacting with and modifying the environment in which it is operating. Second, an agent is autonomous, so it is not directly controlled by the user but acts in accordance with a set of individual tendencies. These tendencies allow the agent to accept or reject goals or rules depending on the current situation. Third, agents have a limited perception of their situation, rather than having global knowledge of the entire environment. Finally, agents may have the ability to reproduce themselves. A genetic algorithm is one common method that allows reproduction of more successful agents while discouraging the reproduction of poor performers (Ferber, 1999).

Using this agent definition, Ferber describes a multi-agent system (MAS) having the following components. First, the MAS is in an environment that will be perceived by agents. From the agent's point of view, the environment is everything except the agent itself. Second, the MAS contains passive objects that can be created, perceived, modified and destroyed by the agents. Third, the MAS contains agents that interact with and operate in the environment. Fourth, a set of relationships is defined for agents and objects to allow communication between them. Fifth, the agents have an associated set of operations that it can perform to interact with other agents, objects or the environment. Finally, the MAS is governed by a set of universal laws that determine how objects and agents respond within the environment (Ferber, 1999).

Agent-based systems have typically been used to model the behavior and interactions of a population or organization. These models have been used to investigate emergent agent or object relationships and group dynamics, such as differentiation or aggregation. As a computational method, agent-based systems offer great flexibility that could be applied to many areas of military simulation. But, because they are only a computational method, the model designer must supply any cognitive validity.

C. TOWARD HYBRID SYSTEMS

As military simulation environments continue to increase in size and complexity, the human performance component of the model is required to more thoroughly represent all aspects of human cognition. But, as discussed above, each of these architectures has different strengths and weaknesses with respect to its ability to accurately represent

various cognitive functions. Given these limitations, it is unlikely that any single architecture could adequately meet all of the cognitive requirements of a simulation environment that approaches real-life complexity. Collectively, however, they provide the building blocks necessary to develop improved models. A long-term solution to these shortcomings focuses on improving and expanding the capabilities of existing approaches, as well as searching for new approaches that can become the basis for entirely new architectures. A near-term solution that attempts to take full advantage of established implementations is to combine the strengths of two or more architectures to produce a hybrid system with a wider range of cognitive capabilities. Hybrid Architectures as Models of Human Learning is an ongoing research program at the Office of Naval Research that has supported the investigation of several initial hybrid systems. In order to support continued development of hybrid systems, the individual architectures that are currently available must be extensively employed and evaluated to more thoroughly establish their relative strengths and weaknesses.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DOMAIN SELECTION AND DESCRIPTION

A. NDM APPROACH APPLIED TO SUBMARINE OPERATIONS

1. Complex Domain

Recall that the NDM domain is characterized by ill-structured problems, a dynamic environment, competing goals, time stress and high stakes. Each of these characteristics plays a part in the decisions made by the Officer of the Deck (OOD) onboard a submarine as he maneuvers the vessel to carry out the assigned mission.

Nearly every aspect of the submerged operating environment is continuously changing. The sonar system's ability to detect other vessels is a function of several variables including background noise from other vessels or marine animals, weather conditions on the surface, water temperature and salinity, water depth, bottom contour, and the relative position of the contact with respect to the submarine. Mission tasking messages may direct a change in the operating area boundaries or even change the mission. As the submarine maneuvers new navigation hazards may be detected. Some of these hazards, such as oil platforms or navigation buoys, will be fixed in one location. If another vessel is detected the OOD must first determine its range, course and speed, and then continue to monitor it to maintain a safe distance. The course, speed and depth of the submarine are also changing as directed by the OOD.

The competing goals that the OOD must prioritize include elements of both safety and mission tasking. His highest goals are to avoid running aground and to avoid colliding with another vessel or stationary navigation hazard. His next highest goal is to remain within the assigned operating area boundaries. Finally, when these safety goals are met, he carries out the assigned mission.

Time stress and high stakes are also very real in the decisions that the OOD is making. In a head-on situation a new contact may close from its initial detection range to the point of collision in a matter of minutes. If an enemy warship is detected the OOD must immediately decide how to maneuver the submarine to avoid being counterdetected and, if necessary, how to employ the weapon systems. High stakes apply not only in the

wartime situation just described, but also in collision and grounding situations where the potential for loss of lives or severe damage to a multi-million dollar warship is very much a reality.

2. Experienced and Knowledgeable Decision Makers

The U.S. Navy's submarine OOD training and qualification process takes newly commissioned officers and produces experienced, knowledgeable decision makers and submarine drivers. The process begins with six months of very technical classroom instruction covering the details of the submarine's nuclear propulsion plant. This is followed by six months of propulsion plant operational training as the Engineering Officer of the Watch (EOOW), supervising a twelve-man propulsion plant watch team. The propulsion plant could also be characterized as a complex NDM domain, and this experience as an EOOW exposes the young officer to a multitude of decision making and watch team coordination scenarios. The next phase is three months of classroom training on basic submarine design and operation, building a foundation for the knowledge required of an OOD. Topics include technical and design aspects of ship control, navigation, sonar and weapon systems, as well as the operational guidance for each system. When the officer reports to a submarine he completes EOOW qualifications first, usually three to four months after reporting. In this role he continues to gain experience making decisions in a complex domain and coordinating the efforts of a watch team. Once he has qualified EOOW, his OOD qualification process begins its final phase. A typical day at sea includes a six-hour EOOW watch period, followed by a six-hour OOD training watch under the supervision of senior, experienced officers. The training watches are critical in that they provide him with exposure to many of the scenarios he will encounter as an OOD, as well the opportunity to discuss the scenarios with more experienced officers. The training watch routine will continue for four to six months. The final step of the OOD qualification includes written and oral examinations to confirm that the officer has a complete technical understanding of various submarine systems, and the ability to make the correct operational decisions in a variety of scenarios

presented to him. Once he is a qualified OOD, he will stand a six hour watch each day at sea, continuing to build his level of knowledge and experience.

3. Situation Assessment

Doctrinal guidance for submarines contains specific procedures that should be followed for various operational and emergency scenarios. These procedures provide the OOD with a list of actions that will either improve the tactical situation, or in the case of an emergency, place the submarine in a safe condition. Given that these procedures exist, the real problem that the OOD must solve is deciding which of the action lists is applicable for the current situation.

B. SUBMARINE OPERATING ENVIRONMENT

1. Operating Area and Tasking

For this research the submarine has been assigned a large rectangular operating area. The operating area contains several stationary obstacles and a contoured bottom. The submarine's tasking is to safely transit from a randomly generated start point to a randomly generated destination point.

2. Sensor Capabilities and Maneuvering Options

Three submarine systems provide the OOD decision models (OODDM) with information about the environment. The sonar system provides information about all obstacles that are within sensing range. The fathometer provides the depth of the shallowest point on the bottom in the immediate vicinity. The navigation system provides the values of current maneuvering parameters (position, course, speed and depth) and the distance to operating area boundaries. Based on this information the OODDM maneuvers the submarine by making changes to its course, speed or depth.

C. OOD DECISION MODEL DESIGN CONSIDERATIONS

To ensure that any differences observed in the performance of the two OODDMs were only due to differences in the actual decision making process, all other domain parameters were held constant. Both OODDMs operate in the same operating area with the same bottom contour. All sonar, fathometer and navigation system information received, and the maneuvering options available to each OODDM are identical. Table 1 below provides a list of the domain parameters that were held constant and their values.

D. IDENTICAL DOMAIN FRAMEWORK

1. Operating Area and Horizontal Coordinate System

A 1000 x 600 pixel grid is used as the basis for the xy-coordinate system. Using a 20 pixels = 1 nautical mile (nm) scale, this results in a 50 nm x 30 nm operating area. The origin ($x = 0$, $y = 0$) of the xy-coordinate system is located in the upper left corner.

2. Bottom Contour and Fathometer Capability

The bottom contour for the operating area was created from a Digital Terrain Elevation Data (DTED) file from the National Imagery and Mapping Agency (NIMA). The DTED file selected was of a mountainous area with several peaks and valleys, with elevations ranging from a minimum of 29 feet to a maximum of 1217 feet. To use this land elevation data as an ocean floor, the ocean surface was set at an elevation of 1400 feet. This resulted in water depths ranging from 183 feet to 1371 feet. The fathometer provides the OODDM with the depth of the shallowest point in a 2 nm x 2 nm square footprint centered on the submarine's current position.

3. Obstacles and Sonar Capabilities

There are 25 stationary obstacles randomly located throughout the operating area. The minimum distance between obstacles is 3 nm. The sonar system provides the OODDM with the position of each obstacle that is within maximum sensing range (5 nm)

of the submarine's current position. The sonar system also designates the obstacles as either critical (0-2.5 nm) or distant (2.5-5 nm) to assist the OODDM in prioritizing them.

4. Starting Position and Destination Point

The submarine's starting position and destination point are based on randomly generated xy-coordinates. These two locations are not allowed to be within critical contact range (2.5 nm) of any obstacles. To allow the submarine to start in a stable condition it is assigned an initial speed of 10 knots and an initial operating depth of one-third of the bottom depth at the starting position. Its initial course is randomly selected from the eight possibilities (N, NW, W, SW, S, SE, E, NE).

5. Course, Speed and Depth Change Options

After evaluating the situation, the OODDM can order small or large changes to each of the operating parameters or leave them unchanged. Course changes can be to the left or to the right in a small (45°) or large (90°) increments. Speed can be increased or decreased by 5 or 10 knots, with minimum and maximum values set at 5 and 25 knots. Depth can be increased or decreased by 25 or 50 feet, with a minimum operating depth of 50 feet and a maximum operating depth of 800 feet.

6. Mission Tasking and Operating Goals

The submarine's assigned task is to safely transit from the starting position to the destination point. Operating goals are as follows: first – avoid grounding, second – avoid collision, third – remain within the operating area, fourth – transit to the destination. Secondary considerations are to conduct the transit as quickly as possible, and to operate as deep as possible.

Operating Area Parameters	
Distance conversion scale	20 pixels = 1 nautical mile (nm)
Area size	50 nm x 30 nm
Bottom contour	The same DTED database was used
Minimum bottom depth	183 feet
Maximum bottom depth	1371 feet
Number of stationary obstacles	25
Minimum distance between obstacles	3 nm
Sonar System Parameters	
Maximum sensing range	5 nm
Critical contact range	2.5 nm
Fathometer System Parameters	
Footprint size	2 nm x 2 nm centered on current position
Red sounding depth	200 feet
Yellow sounding depth	400 feet
Navigation System Parameters	
Warning distance to boundary area	5 nm
Critical distance to area boundary	2.5 nm
Maneuvering Parameters	
Course options	E, NE, N, NW, W, SW, S, SE
Course change options	No change, 45° left or right, 90° left or right
Speed change options	No change, +/- 5 knots, +/- 10 knots
Minimum speed	5 knots
Maximum speed	25 knots
Depth change options	No change, +/- 25 feet, +/- 50 feet
Minimum operating depth	50 feet
Maximum operating depth	800 feet

Table 1. Constant Domain Parameters

IV. AGENT-BASED OOD DECISION MODEL IMPLEMENTATION

A. DESCRIPTION AND DESIGN

In Chapter III it was pointed out that doctrinal operating guidance provides a list of actions to be taken for various tactical and emergency scenarios. The problem that the submarine OOD must solve then is not deciding what maneuvering commands he should order. Instead, the problem is to determine, based on his current situation, which list of actions is applicable. Using this as its foundation, each cycle of the agent-based OODDM consists of three steps. First, current information is received from the sensor systems. Second, operating goals are considered to determine which of the doctrinal scenarios best fits the current situation. Third, predetermined actions are taken for that scenario.

B. MANEUVERING ACTION SETS

Maneuvering action sets are used to represent all possible combinations of course, speed and depth changes available to the OODDM. Each of the maneuvering parameters can be left unchanged, or adjusted in a small or large increment. This results in a total of 27 different maneuvering action sets.

1. Critical Action Sets

Critical action sets are available to the OODDM when the current situation is determined to be in extremis with regard to any of the operating goals. The process for making this determination is fully described later in this chapter. To allow the OODDM every possible option when maneuvering to improve this situation, all 27 maneuvering action sets are included as critical action sets. Table 2 lists the critical action sets.

ACTION SET	COURSE CHANGE	SPEED CHANGE	DEPTH CHANGE
0	None	None	None
1	None	None	25 ft
2	None	None	50 ft
3	None	5 kts	None
4	None	5 kts	25 ft
5	None	5 kts	50 ft
6	None	10 kts	None
7	None	10 kts	25 ft
8	None	10 kts	50 ft
9	45°	None	None
10	45°	None	25 ft
11	45°	None	50 ft
12	45°	5 kts	None
13	45°	5 kts	25 ft
14	45°	5 kts	50 ft
15	45°	10 kts	None
16	45°	10 kts	25 ft
17	45°	10 kts	50 ft
18	90°	None	None
19	90°	None	25 ft
20	90°	None	50 ft
21	90°	5 kts	None
22	90°	5 kts	25 ft
23	90°	5 kts	50 ft
24	90°	10 kts	None
25	90°	10 kts	25 ft
26	90°	10 kts	50 ft

Table 2. Critical Action Sets

2. Moderate Action Sets

Moderate action sets are available to the OODDM when the current situation is determined to be less than critical. Again, the process for making this determination is fully described later in this chapter. In these situations the OODDM is not allowed to make large changes to maneuvering parameters. Table 3 lists the moderate action sets.

ACTION SET	COURSE CHANGE	SPEED CHANGE	DEPTH CHANGE
0	None	None	None
1	None	None	25 ft
2	None	5 kts	None
3	None	5 kts	25 ft
4	45°	None	None
5	45°	None	25 ft
6	45°	5 kts	None
7	45°	5 kts	25 ft

Table 3. Moderate Action Sets

3. Active Action Sets

Active action sets are used to represent the lists of predetermined actions from doctrinal guidance. Recall that there are four operating goals – avoid grounding, avoid collision, remain within in the operating area, and transit to the destination point. The first three goals are each assigned two active action sets, one to address critical situations and one to address moderate situations. The transit goal is only assigned a moderate active action set. The programmer initially selects the seven active action sets. The OODDM also provides recommendations on the selection of future active action sets based on evaluations of the performance of each maneuvering action set. This learning process is fully described later in this chapter. Table 4 lists the initial active action sets.

GOAL SCENARIO	ACTIVE ACTION SET	COURSE CHANGE	SPEED CHANGE	DEPTH CHANGE
Grounding (critical)	Critical Action Set 26	90°	10 kts	50 ft
Grounding (moderate)	Moderate Action Set 3	None	5 kts	25 ft
Collision (critical)	Critical Action Set 24	90°	10 kts	None
Collision (moderate)	Moderate Action Set 6	45°	5 kts	None
Area (critical)	Critical Action Set 24	90°	10 kts	None
Area (moderate)	Moderate Action Set 6	45°	5 kts	None
Transit (moderate only)	Moderate Action Set 7	45°	5 kts	25 ft

Table 4. Initial Active Action Sets

C. DECISION PROCESS

Appendix A contains pseudocode of the decision process.

1. Setting Operating Goal Weights

The OODDM sets the “weight” of each operating goal as either critical or moderate based on current sensor system information. The following criteria are used for each operating goal:

a. Avoid Grounding Goal

The avoid grounding goal weight is set to critical when the difference in depth between the submarine’s keel and the ocean bottom is less than an established minimum value. This minimum distance, called a red sounding, has been set at 200 feet. Similarly, the avoid grounding goal weight is set to moderate when the distance beneath the keel is not critical, but is less than the yellow sounding value, which has been set at

400 feet. The depth difference is calculated by subtracting the submarine's current operating depth from the fathometer system input, which reports the shallowest point in a 2 nm x 2 nm square footprint centered on the submarine's position.

b. Avoid Collision Goal

The avoid collision goal weight is set to critical when any obstacle is within critical contact range (2.5 nm). It is set to moderate when any obstacle is within maximum sensing range (5 nm), but is outside of critical contact range. The range to contacts is provided by the sonar system and is calculated using the xy-coordinates of the obstacle's position and the submarine's position.

c. Remain Within Operating Area Goal

The remain within operating area goal weight is set to critical when the submarine closes to less than a preset critical distance (2.5 nm) from any area boundary. It is set to moderate when the submarine is less than a preset warning distance (5 nm), but greater than the critical distance from any area boundary.

d. Transit to Destination Goal

The transit to destination goal weight is always set to moderate.

2. Active Action Set Selection and Conflict Resolution

The OODDM uses the operating goal weights to determine which of the seven active action set scenarios is appropriate for the current situation. The OODDM considers critical operating goals first. If only one of the operating goal weights is critical, the active action set corresponding to that scenario is selected. If more than one operating goal weight is critical, the OODDM uses operating goal priorities to resolve the conflict. If there are no critical operating goals, the OODDM considers moderate operating goals. The transit to destination goal weight is always moderate, so there will always be at least one moderate operating goal.

3. Maneuvering Commands

Selecting the active action set only determines the magnitude of course, speed and depth changes. The OODDM must also determine the direction of each maneuvering parameter change for each operating goal.

a. Avoid Grounding Goal

Course is changed to the right to reverse directions and drive back toward safe water. Speed is reduced to minimize closure to shallow point. Operating depth is reduced to maximize depth beneath the keel.

b. Avoid Collision Goal

If the obstacle is directly ahead or to the left of the submarine's current heading, then course is changed to the right. If the obstacle is to the right of the submarine's current heading, then course is changed to the left. Speed is reduced to minimize closure to the obstacle. No depth change is required.

c. Remain Within Operating Area Goal

If the operating area boundary is directly ahead or to the left of the submarine's current heading, then course is changed to the right. If the operating area boundary is to the right of the submarine's current heading, then course is changed to the left. Speed is reduced to minimize closure to the boundary. No depth change is required.

d. Transit to Destination Goal

If the destination point is to the left of the submarine's current heading, then course is changed to the left. If the destination point is to the right of the submarine's current heading, then course is changed to the right. Speed is increased to minimize the time required to reach the destination point. Operating depth is increased to meet the secondary consideration of operating as deep as possible.

D. LEARNING PROCESS

The OODDM gathers information during each decision cycle that allows it to compute performance evaluations for each of the 27 critical action sets and eight moderate action sets at the end of the mission. These evaluations can be used to assign better performing active action sets to the seven possible goal scenarios.

1. Action Set Performance Evaluation

During each decision cycle the OODDM applies a scoring function to each of the action sets that are available for the current goal scenario. The score value gives an indication of how the tactical situation would have changed if the maneuvers of that action set had been taken. At the end of the mission the scores for each action set are totaled, allowing the user to compare the performance of each action set with the current active action sets. To evaluate each action set, first the maneuvers are simulated, and then the scoring function is applied to determine how much the tactical scenario has improved or worsened. The scoring function for each operating goal is described below.

a. Avoid Grounding Goal

Water depth beneath the keel is metric for this scoring function, with better performance shown by larger values. The score value is calculated by subtracting the submarine's operating depth from the fathometer input, both based on the new simulated position of the submarine.

b. Avoid Collision Goal

Distance to the nearest obstacle is the metric for this scoring function, with better performance shown by larger values. The score value is equal to the distance to the nearest obstacle, based on the submarine's position after the simulated maneuver.

c. Remain Within Operating Area Goal

Distance to the nearest operating area boundary is the metric for this scoring function, with better performance shown by larger values. The score value is

equal to the distance to the nearest boundary, based on the submarine's position after the simulated maneuver.

d. Transit to Destination Goal

Distance to the destination point is the metric for this scoring function, with better performance shown by smaller values. The score value is equal to the distance to the destination point, based on the submarine's position after the simulated maneuver.

2. Assigning New Active Action Sets

At the end of the mission the performance scores are written to an output file. This output shows how each of the 27 action sets would have performed for each of the four operating goals. The user can then review the action set performance for each operating goal scenario and manually assign new active action sets as desired. Assigning new active action sets could easily be programmed to occur automatically at the end of each mission. To meet the model comparison needs of this research, complete control of active action sets by the user was desired, so automatic assignment was not implemented.

V. RULE-BASED OOD DECISION MODEL IMPLEMENTATION

A. DESCRIPTION AND DESIGN

The Java™ Expert System Shell (JESS) was selected as the foundation for the rule-based OODDM implementation. Several factors contributed to JESS's selection for this research. Most importantly, since JESS is written entirely in Java™ it can easily be integrated into, or used in conjunction with, a myriad of existing systems. Also, as discussed in Chapter II, the JESS inference engine uses the Rete algorithm and is extremely efficient. User support is readily available as well. Downloadable trial software and documentation, answers to frequently asked questions, and points of contact are available at the JESS website (<http://herzberg.ca.sandia.gov/jess/>).

Similar to the agent-based OODDM, each cycle of the JESS OODDM consists of three steps. First, current sensor information is provided by asserting the information as facts that are added to the knowledge base. Second, the inference engine determines which of the rules are appropriate for the situation. The rules represent the four operating goals (avoid grounding, avoid collision, remain within the operating area, and transit to destination). Third, maneuvering commands are executed as the rules fire.

B. KNOWLEDGE BASE FACTS

1. Fact Templates

A fact template defines the format of the information contained in a fact. The template names the fact and creates a “slot” for each desired fact characteristic. JESS code for a typical fact template and an associated fact is shown below.

```
(deftemplate automobile (slot make) (slot model) (slot year) (slot color) )
```

```
(automobile (make Ford) (model Mustang) (year 1997) (color red) )
```

Several fact templates have been defined to allow the operating parameters and sensor system information associated with the OODDM to be asserted in the knowledge base. Figure 1 below shows the OODDM fact templates in JESS format.

a. *Status Fact Template*

This fact is used to store the submarine's position and operating parameter information. It has slots for x and y coordinates, course, speed and depth.

b. *Destination Fact Template*

This fact is used to store the position of the destination point and its distance and direction from the submarine's current position. It has slots for x and y coordinates, distance and direction.

c. *Obstacle Fact Template*

This fact is used to store information provided by the sonar system for all obstacles within sensing range. It has slots for x and y coordinates, distance and direction.

d. *Shoal Fact Template*

This fact is used to store fathometer system information. It has slots for minimum depth and sounding values. Minimum depth is the shallowest point in a 2 nm x 2 nm footprint centered on the submarine's position. Sounding values can be red, yellow or safe.

e. *OutArea Fact Template*

This fact is used to store information when the submarine approaches an operating area boundary. It has slots to indicate the boundary of concern (top, bottom, left or right) and the distance to that boundary.

(deftemplate status (slot x)(slot y)(slot course)(slot speed)(slot depth))
(deftemplate destination (slot x)(slot y)(slot distance)(slot direction))
(deftemplate obstacle (slot x)(slot y)(slot distance)(slot direction))
(deftemplate shoal (slot minDepth)(slot sounding))
(deftemplate outArea (slot side)(slot distance))

Figure 1. JESS Fact Templates

2. Asserting Facts

As the first step in each decision cycle, facts are asserted to store current operating parameters and sensor system information in the knowledge base. Navigation system information is used to assert three types of facts – status, destination and outArea. Sonar system information is used to assert obstacle facts for all obstacles that are within maximum sensing range. Fathometer system information is used to assert shoal facts.

C. KNOWLEDGE BASE RULES

JESS rules are used to specify a set of actions to be performed for a given situation. A rule is composed of an *if* portion and a *then* portion. The *if* portion of a rule is a series of facts that must be satisfied to cause the rule to be applicable. The *then* portion of a rule is the set of actions to be executed when the rule is applicable.

JESS code for a simple rule is shown below. The rule name is “do-change-baby.” If the fact “baby-is-wet” is true, then the action “change-baby” will be taken.

```
(defrule do-change-baby
  (baby-is-wet)
  =>
  (change-baby))
```

For the OODDM, rules are used to represent each of the four operating goals. The facts satisfied in the *if* portion determine which of the seven operating goal scenarios exists. The actions taken in the *then* portion assign the corresponding course, speed and depth changes. For consistency with the agent-based OODDM, the parameter values listed in Table 1 were again used to determine the level of urgency of the situation. Table 5 shows the course, speed and depth changes for each operating goal scenario. Note that the values are the same as those used for the agent-based OODDM’s initial active action sets (Table 4).

GOAL SCENARIO	COURSE CHANGE	SPEED CHANGE	DEPTH CHANGE
Grounding (critical)	90°	-10 kts	-50 ft
Grounding (moderate)	None	-5 kts	-25 ft
Collision (critical)	90°	-10 kts	None
Collision (moderate)	45°	-5 kts	None
Area (critical)	90°	-10 kts	None
Area (moderate)	45°	-5 kts	None
Transit (moderate only)	45°	+5 kts	+25 ft

Table 5. Maneuvering Parameter Changes For Operating Goal Scenarios

1. Avoid Grounding Goal Rules

The shoal fact contains a sounding slot that has been assigned a value of RED (fathometer system reports less than 200 feet) or YELLOW (fathometer system reports less than 400 feet). The rule representing the grounding critical scenario simply checks to see if a shoal fact with a sounding value of RED exists, and assigns the associated maneuvering parameter changes if satisfied. Similarly, the rule representing the grounding moderate scenario checks for a shoal fact with a sounding value of YELLOW.

2. Avoid Collision Goal Rules

The obstacle fact contains distance and direction slots. Two rules are required to represent the collision critical scenario to allow the OODDM to select either a right or left turn. The first rule checks for the existence of an obstacle fact with a distance less than critical contact range (2.5 nm) and a direction to the left of or equal to the submarine's current course. If these conditions are satisfied the actions are taken for a

turn to the right. The second rule checks for the existence of an obstacle fact with a distance less than critical contact range (2.5 nm) and a direction to the right of the submarine's current course. If these conditions are satisfied the actions are taken for a turn to the left. Two similar rules represent the collision moderate scenario to address obstacles that are within maximum sensing range (5 nm), but outside of critical contact range.

3. Remain Within Operating Area Goal Rules

The outArea fact contains a side slot that indicates the boundary area of concern (top, bottom, left or right) and a distance slot. Two rules are again required to represent the area critical scenario to allow the OODDM to select either a right or left turn. The first rule checks for the existence of an outArea fact with a distance less than critical boundary range (2.5 nm). Based on the value of the side slot, the rule checks to see if the boundary is to the left of or equal to the submarine's current course. If these conditions are satisfied the actions are taken for a turn to the right. The second critical scenario rule checks for the existence of an outArea fact with a distance less than critical boundary range (2.5 nm) and the boundary positioned to the right of the submarine's current course. If these conditions are satisfied the actions are taken for a turn to the left. Two similar rules represent the area moderate scenario to address boundaries that are within warning range (5 nm), but outside of critical boundary range.

4. Transit to Destination Goal Rules

Two rules are again required to represent the transit scenario to allow the OODDM to select either a right or left turn. Based on the value of the direction slot, the first rule checks to see if the destination is to the left of the submarine's current course and, if this condition is satisfied the actions are taken for a turn to the right. A similar rule addresses transit situations requiring a turn to the left. To meet secondary considerations of minimizing transit time and operating as deep as possible, these rules also assign speed and depth increases, as indicated in Table 5.

D. OPERATING GOALS AND CONFLICT RESOLUTION

Recall that the operating goal priorities are, from highest to lowest, to avoid grounding, to avoid collision, to remain in the operating area, and to transit to the destination. A two-part process that uses scenario tracking facts and rule salience was implemented to ensure that these priorities are met.

1. Scenario Tracking Facts

In addition to assigning course, speed and depth changes, the *then* portion of each goal scenario rule also asserts a scenario tracking fact to indicate that conditions are satisfied for that scenario. This allows each goal scenario rule to verify that conditions are not met for a higher operating priority. Consider the following example. An obstacle is within critical contact range and, at the same time, the submarine is within critical boundary distance of an area boundary. To meet operating goal priorities actions should be taken to avoid collision. Because the conditions are met for the collision critical scenario an “avoiding-close-obstacle” scenario tracking fact is asserted. Conditions for the area critical rule will not be satisfied because it checks that scenario tracking facts do not exist for higher priority operating goals.

2. Scenario Rule Salience

The scenario tracking fact policy will not work if the conditions for a low priority scenario rule are satisfied before the higher priority rule has asserted the tracking fact. To control the order in which they are processed, each scenario rule is assigned a salience value. Rules are processed in order from highest to lowest salience value. This combination of scenario rule salience and scenario tracking facts ensures that operating goal priorities are met.

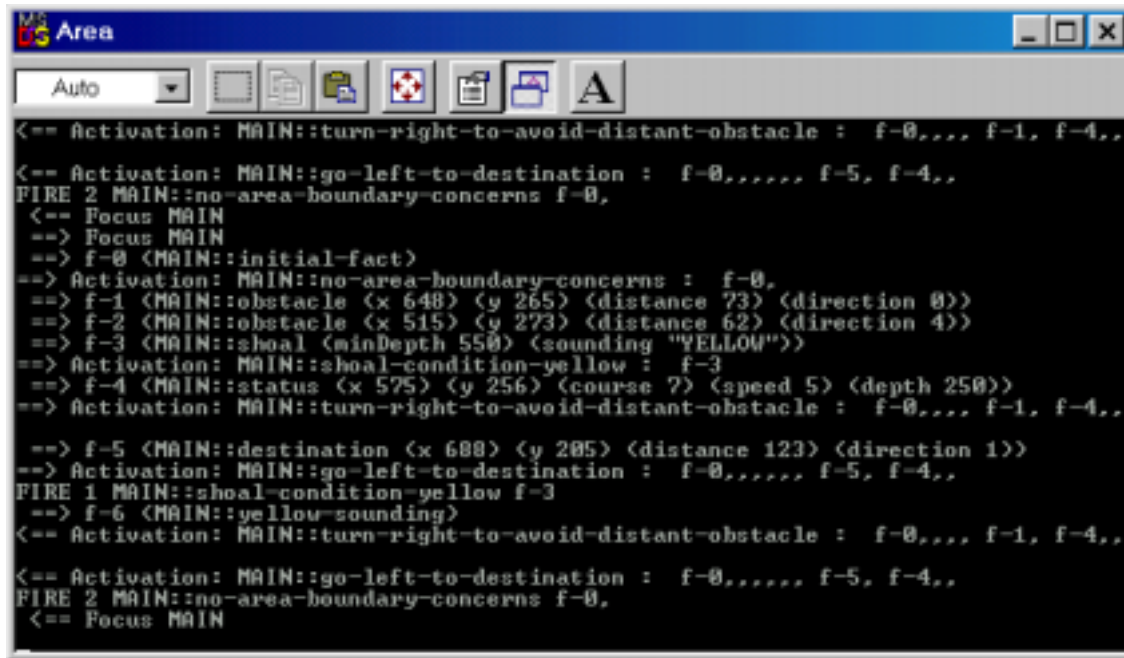
E. EXPLAINING OODDM ACTIONS

1. The JESS Watch Function

The designers of JESS developed the “watch” function as a debugging tool to assist the user in troubleshooting rules that are not operating as expected. The watch function allows the user to monitor the execution of the Rete inference engine by providing information concerning the status of facts and rules. For facts, it reports every time a fact is asserted or retracted. A fact is retracted when it is no longer true. For rules, the watch function reports every time a rule is activated or deactivated, and lists the facts that caused the event to occur. A rule is activated when all conditions of the rule’s *if* portion are satisfied. If the operating environment changes in such a way that a required fact is retracted the *if* portion is no longer satisfied, and the rule will be deactivated. The watch function also reports every time a rule fires (carries out the actions of the *then* portion) and lists the facts that were used to satisfy the rule’s conditions.

2. OODDM Application of the Watch Function

While the watch function is certainly a useful debugging tool, it can be used to perform an even more important function for the OODDM. By allowing the watch function to operate continuously it provides the user with information that shows why the OODDM made each decision. This information can be separated into three distinct parts. First, the assertion and retraction of facts represents the OODDM’s perception of the operating environment, as reported by the sensor systems. Second, the activation and deactivation of rules represent the maneuvering options being considered, but not yet chosen. Finally, the rules fired represent the OODDM’s final decisions. The output of the watch function can either be displayed in a text window, so that it can be reviewed in real time as the OODDM is running, or it can be stored as a data file and reviewed after the OODDM completes the entire mission. Figure 2 shows a DOS window containing the reports of the watch function while the OODDM is operating.



```
<== Activation: MAIN::turn-right-to-avoid-distant-obstacle : f-0,,, f-1, f-4,,
<== Activation: MAIN::go-left-to-destination : f-0,,,,, f-5, f-4,,
FIRE 2 MAIN::no-area-boundary-concerns f-0,
<== Focus MAIN
==> Focus MAIN
==> f-0 <MAIN::initial-fact>
==> Activation: MAIN::no-area-boundary-concerns : f-0,
==> f-1 <MAIN::obstacle <x 648> <y 265> <distance 73> <direction 0>>
==> f-2 <MAIN::obstacle <x 515> <y 273> <distance 62> <direction 4>>
==> f-3 <MAIN::shoal <minDepth 550> <sounding "YELLOW">>
==> Activation: MAIN::shoal-condition-yellow : f-3
==> f-4 <MAIN::status <x 575> <y 256> <course 7> <speed 5> <depth 250>>
==> Activation: MAIN::turn-right-to-avoid-distant-obstacle : f-0,,, f-1, f-4,,

--> f-5 <MAIN::destination <x 688> <y 285> <distance 123> <direction 1>>
--> Activation: MAIN::go-left-to-destination : f-0,,,,, f-5, f-4,,
FIRE 1 MAIN::shoal-condition-yellow f-3
--> f-6 <MAIN::yellow-sounding>
<== Activation: MAIN::turn-right-to-avoid-distant-obstacle : f-0,,, f-1, f-4,,

<== Activation: MAIN::go-left-to-destination : f-0,,,,, f-5, f-4,,
FIRE 2 MAIN::no-area-boundary-concerns f-0,
<== Focus MAIN
```

Figure 2. JESS Watch Function Reports

VI. OOD DECISION MODEL COMPARISON

A. COMPARISON PLAN

First, the graphical interface used to display the operating area and control the operation of the OODMs is fully described. The following comparison areas are then addressed: implementation considerations, performance and observed behavior, cognitive function representation, and cognitive validity.

B. GRAPHICAL INTERFACE

1. Display Window

A 1000 x 600 pixel window is used to display the submarine's movement through its operating area. The horizontal xy-coordinate grid uses the upper left (northwest) corner as its origin (0,0) point. The window's borders represent the submarine operating area boundaries. Obstacles in the area are shown as black stars, and the destination point as a blue star. The submarine is shown as a square that varies in color depending on sounding conditions. For red (< 200 feet beneath the keel) or yellow (< 400 feet beneath the keel) sounding conditions, the submarine is colored red or yellow, respectively. If the sounding condition is safe (> 400 feet beneath the keel) the submarine is colored green.

2. Operating Data Dialog Box

The display window only provides a top-down view of the submarine's horizontal movement in the operating area. An operating data dialog box is used to provide the user with depth information. Clicking the mouse on any of the obstacles or the destination point activates a dialog box that displays the bottom depth and xy-coordinates of that location. Clicking the mouse on the submarine will display its position, operating depth and the bottom depth. Figure 3 shows the display window with an activated operating data dialog box for the submarine.

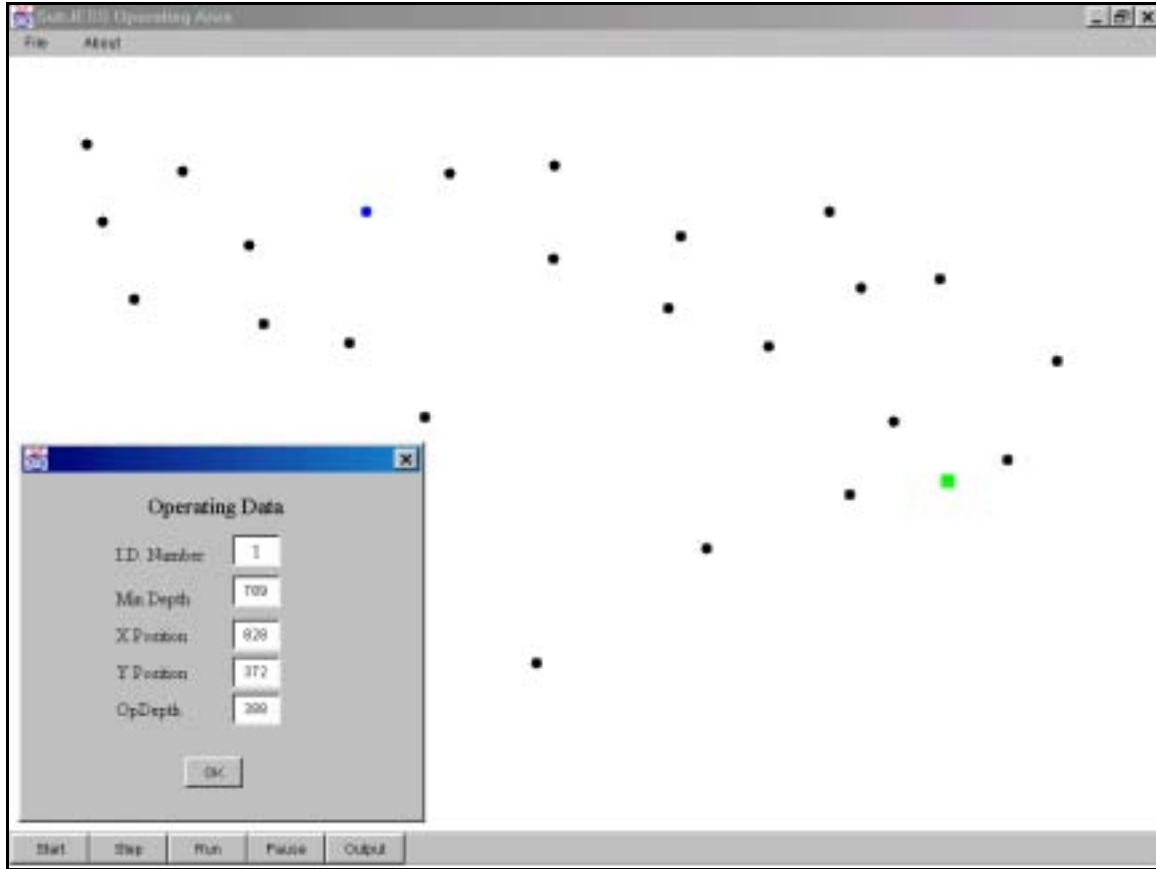


Figure 3. Operating Area Display Window

3. Control Buttons

Five buttons are used to control the environment and OODDM. The START button fills the empty operating area. It creates the area's bottom contour by loading the elevation data from the DTED file. It also randomly positions the obstacles, randomly generates the destination point, and establishes the submarine's starting conditions. The STEP button causes one decision cycle to be performed. The RUN button allows the OODDM to run until the submarine arrives at the destination point or the PAUSE button is pressed. The PAUSE button stops execution until the STEP or RUN button is pressed to resume execution. The OUTPUT button writes the submarine's path history to an output file. The path history consists of the submarine's position, course, speed and depth at each decision cycle of the mission. For the agent-based OODDM, the OUTPUT button also writes the action set evaluation scores to an output file.

4. DOS Display Window

A DOS window is used to display textual information during execution. For the rule-based OODDM, the DOS window is used to display the output of the watch function as it reports assertion and retraction events for facts, and activation, deactivation and firing events for rules during each decision cycle. For the agent-based OODDM, the DOS window is used to display the action set evaluation scores when the OUTPUT button is pressed at the end of the mission.

C. IMPLEMENTATION CONSIDERATIONS

Given that both OODDMs were written in Java™, several well-known Java™ features were apparent throughout this research. These included its compatibility and portability, extensive user group and documentation available for assistance, ease of file management, and graphical display development framework. Implementation considerations for each OODDM are discussed in the following sections.

1. Agent-Based OODDM

In order to implement the learning process of the agent-based OODDM, critical action sets (Table 2) and moderate action sets (Table 3) were required to specify every possible combination of maneuvering parameter changes for a given operating goal scenario. Then the OODDM simply steps through each action set, calculates a performance score for the current operating goal, and adds the score to the action set's running total for the entire mission. At the end of the mission the performance score total for each action set can be evaluated to determine which action sets would have been most effective for each operating goal. A second benefit of the action set system is that it lays the groundwork for the next level of agent-based OODDM improvement - the genetic algorithm - which is discussed more fully in the future work section of Chapter VII.

2. Rule-Based OODDM

There are two key features of JESS that make it an ideal choice for use in the development of a submarine OODDM. First, the knowledge base of facts and rules parallels directly with sensing system inputs (asserted as facts) and doctrinal guidance (implemented as rules), making rule writing fairly straight forward. Second, JESS's watch function provided an easily interpreted realtime output showing why the OODDM was making each decision by showing what conditions were satisfied and which rules were fired.

The major disadvantage of using a production engine like JESS is that each and every rule must be written by the programmer. This means that either the programmer must have a very good understanding of the OODDM environment and decision process, or someone else must provide very detailed guidance to him. In either case, the OODDM will only be as good as the rules that are written for it. As the complexity of the environment increases the task of writing a complete, correct rule set becomes even more difficult.

A second JESS-specific problem is that not all CLIPS functions have been implemented in the current version of JESS. This limitation forces the programmer to write some portions of the code in CLIPS (Lisp based) files that are read by JESS.

D. PERFORMANCE AND OBSERVED BEHAVIOR

Given the environmental and sensing system parameters that were held constant (Table 1) and the predetermined maneuvering commands for the seven operating goal scenarios (Tables 4 and 5), the performance of the two OODDMs was expected to be nearly identical. The only noticeable differences in behavior occurred in situations where the OODDMs were presented with several obstacles in the same range (critical or moderate). These differences were caused by differences in the implementation of obstacle conflict resolution methods. The agent-based OODDM resolves this conflict based on distance, taking action to avoid the nearest obstacle. The rule-based OODDM

takes action based on the order in which the facts were asserted, regardless of the distance to each obstacle.

E. COGNITIVE FUNCTION REPRESENTATION

Recall from the cognitive architecture reviews in Chapter II that agent-based systems and JESS are not based on specific unified theories of cognition. Instead, they are better described as frameworks, or shells, for designing models of human decision making or human behavior. Because of this the programmer must implement any representations of human cognitive functions. The following cognitive functions are represented in the OODDMs: sensing and perception, long and short-term memory, multi-tasking ability, decision making, learning (agent-based OODDM only), and the ability to explain actions (rule-based OODDM only). The following sections describe how each of these functions is represented in each OODDM.

1. Sensing and Perception

Due to the nature of submerged operations, the sensing and perception function of both OODDMs is strictly limited to the data received from the sensing systems (sonar, fathometer and navigation). The constant domain parameters of Table 1 were used to ensure that both OODDMs “see” the environment the same way.

2. Long and Short-Term Memory

For the agent-based OODDM, long-term memory is represented by the active action sets that contain the maneuvering commands for each of the seven operating goal scenarios. Operating goal priorities (avoid grounding, avoid collision, remain in area and transit) are included in long-term memory as methods written to ensure the priorities are met in the proper order. Its short-term memory is represented by the vectors that contain obstacles and area boundaries within ranges of concern. Short-term memory also includes current operating parameters (position, course, speed, operating depth and

bottom depth) that are stored as submarine object variables. All short-term memory data is refreshed on every decision cycle.

For the rule-based OODDM, long-term memory is represented by the knowledge base rules used by the Rete inference engine. The rules employ scenario tracking facts and salience to “remember” operating goal priorities. Its short-term memory is represented by the assertion and retraction of facts containing information about obstacles, area boundaries, bottom depth and the submarine’s current status. The information contained in each type of fact was shown in Figure 1. New facts are asserted on each decision cycle.

3. Multi-Tasking Ability and Decision Making

For the agent-based OODDM, multi-tasking between the four operating goals is accomplished through a two-step process. First, each operating goal is assigned a goal weight (critical, moderate or nominal) based on sensing system information stored in short-term memory. Second, methods are called to perform conflict resolution and meet operating goal priorities when two or more operating goals have been assigned the same goal weight.

For the rule-based OODDM, the asserted facts may initially satisfy the *if* portion of several operating goal rules. The Rete inference engine resolves this through a combination of scenario tracking facts and rule salience to ensure that the rule associated with the highest priority operating goal is fired.

4. Learning (Agent-Based OODDM Only)

The agent-based OODDM’s learning process uses operating goal scoring functions to evaluate the performance of all 27 critical action sets and eight moderate action sets on each decision cycle. The performance scores are summed over the entire mission and can be used to determine which of the active action sets would have been most effective. In the current implementation the user manually enters the updates to the active action sets. This process could easily be modified to allow the OODDM to

automatically update the active action sets. The learning process was fully described in Chapter IV.

5. Ability to Explain Actions (Rule-Based OODDM Only)

The rule-based OODDM activates the JESS watch function to monitor and display the assertion and retraction of facts, and the activation, deactivation and firing of rules. The user can interpret this information to determine what conditions existed at each decision cycle and explain why the OODDM made the corresponding decision. This ability was described in more detail in Chapter V.

F. COGNITIVE VALIDITY

The validity of a computational cognitive model is evaluated based on the model's design goals. Some models are designed with the goal of implementing the human cognitive functions in a way that accurately represents the psychological elements of an accepted mental model or unified theory of cognition. On the other hand, many models are designed with the goal of producing realistic human behavior with no requirement for a cognitive theory foundation. The OODDMs of this research fall into the second category and the following question is considered to evaluate their cognitive validity. Does the OODDM produce behavior that is representative of the actions that would have been taken by an experienced submarine OOD in the same situation? Based on an informal analysis of each OODDM conducting several transit missions both models exhibited behavior that was expected and believable.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND RECOMMENDATIONS

A. COMPARISON CONCLUSIONS

The purpose of this thesis was to continue researching the development of hybrid computational cognitive architectures by comparing the implementation and performance of two models designed to operate in the same problem environment. From the model comparisons made in Chapter VI, the following conclusions should be noted:

1. Environment Complexity Can Overwhelm

The relatively simple level of complexity of the OODDM's submerged operating environment allowed both models to employ the same basic algorithm:

- Receive new data from sensing systems
- Determine which operating goal scenario exists
- Make maneuvering parameter changes for that scenario
- Move submarine to new position
- Repeat

Even though the implementations were completely different – Java™ methods for the agent-based OODDM and JESS rules for the rule-based OODDM – both models operated by matching the current situation with one of the operating goal scenarios. If the complexity of the environment is increased, the number of operating goal scenarios required to describe possible situations would also increase. More operating goal scenarios will make the matching process more difficult, requiring the creation of more elaborate methods or rules. The number of methods or rules required, and the complexity of those methods or rules, can become very difficult to manage for even moderately complex operating environments.

2. Learning is Powerful

The learning process of the agent-based OODDM is a very powerful capability. As it is currently implemented, at the completion of each mission the OODDM provides a performance evaluation score for each of the available action sets for each operating goal scenario. These scores can be used to determine the most effective action sets, and new active action sets can be assigned by the user if desired. Evaluation of action set performance becomes even more valuable as the complexity of the environment increases. The user must assign the actions to be taken for more detailed operating goal scenarios, and it is unlikely that the most effective actions will be selected without some type of performance feedback.

If the simulation requires an OODDM that “learns as it goes,” the performance scores could be used to automatically assign the most effective action sets as the active action sets at specified intervals throughout the mission.

3. Explanations Must be Useful

Like the agent-based OODDM’s learning process, the rule-based OODDM’s ability to explain its actions is also a powerful capability. But, unfortunately, its usefulness diminishes as the complexity of the operating environment increases. From the description of JESS’s watch function in Chapter V, recall that it reports all activations and firings of rules, and the facts associated with each event. A typical watch function report window was shown in Figure 2. For the OODDM’s submerged operating environment, with its limited number of facts and rules, the user could interpret the watch function output fairly easily. As the number of facts and rules increases, it becomes much less likely that the user will be able to interpret the report directly because he cannot be expected to remember the structure of each fact or rule. This problem can be partially alleviated by using very descriptive fact and rule names, but a complete understanding of the report will require referencing the rules themselves.

B. RECOMMENDATIONS FOR FUTURE WORK

This research could be expanded in several directions to continue supporting the requirements for improved human performance models. Some possibilities for future work include:

1. More Complex Environment or Goals

A more realistic and challenging operating environment could include obstacles that were not stationary, submerged obstacles, or operating area boundaries that changed with time. Operating goal complexity could be increased by shifting the destination point during the transit or by assigning more difficult mission tasking, such as following one of the moving obstacles.

2. Improved Agent-Based OODDM

To take full advantage of an agent-based model's ability to adapt and improve its performance a genetic algorithm could be implemented. The genetic algorithm considers various combinations of sensing system inputs and action set outputs, developing OODDMs that take action based on the most effective input/output combinations. A thorough discussion of genetic algorithm implementation can be found in Ferber's *Multi-Agent Systems*.

3. Improved Rule-Based OODDM

As the environment becomes more complex, and the number of facts and rules increases, the rule-based OODDM's performance could be improved by employing the backward chaining function of JESS's Rete inference engine. When backward chaining is activated, the rule compiler notes which conditions of the *if* portion of a rule are not satisfied. The compiler then generates new rules that show a "need" for the missing conditions, and checks to see if facts exist to satisfy the new rules. Backward chaining is also commonly referred to as goal seeking.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. AGENT-BASED OODDM PSEUDOCODE

1. SENSING SYSTEM INPUTS

The OODDM makes decisions based on inputs from the three sensing systems. The fathometer system provides the minimum bottom depth in a footprint (2 nm x 2 nm) centered on the submarine's position. The navigation system provides the submarine's position (xy-coordinates), course, speed and operating depth. The sonar system provides the locations (xy-coordinates) and distances to all obstacles within maximum sensing range. In the following sections, sensing system inputs and constant domain parameters (Table 1) will be capitalized and pseudocode will be italicized.

2. SETTING OPERATING GOAL WEIGHTS

The first step is to establish goal weights for each of the four operating goals.

If (MIN DEPTH – OP DEPTH) < RED SOUNDING

set Grounding Weight to CRITICAL

If RED SOUNDING < (MIN DEPTH – OP DEPTH) < YELLOW SOUNDING

set Grounding Weight to MODERATE

If (MIN DEPTH – OP DEPTH) > YELLOW SOUNDING

set Grounding Weight to NOMINAL

If any OBSTACLE DISTANCE < CRITICAL CONTACT RANGE

set Collision Weight to CRITICAL

If no OBSTACLE DISTANCE < CRITICAL CONTACT RANGE, but some OBSTACLE

DISTANCE < MAX SENSING RANGE set Collision Weight to MODERATE

If no OBSTACLE DISTANCE < MAX SENSING RANGE

set Collision Weight to NOMINAL

If CURRENT POSITION < CRITICAL DISTANCE from any area boundary

set Area Weight to CRITICAL

If CRITICAL DISTANCE < CURRENT POSITION < WARNING DISTANCE from any area boundary set Area Weight to MODERATE

If CURRENT POSITION > WARNING DISTANCE from any area boundary

set Area Weight to NOMINAL

Always set Transit Weight to MODERATE

3. ACTION SET SELECTION AND CONFLICT RESOLUTION

These operating goal weights are used to select the appropriate active action set from the seven available (Table 4). Recall that the operating goals are, in order of priority, to avoid grounding, avoid collision, remain within operating area boundaries, and transit to the destination.

Consider all CRITICAL goal weights:

If only one goal weight is CRITICAL, then it is selected as the active goal

If more than one goal weight is CRITICAL, the highest priority operating goal is selected as the active goal

If no goal weights are CRITICAL, consider all MODERATE goal weights:

If only one goal weight is MODERATE, then it is selected as the active goal

If more than one goal weight is MODERATE, the highest priority operating goal is selected as the active goal

Note that since the transit goal weight is always set to MODERATE, there will always be at least one MODERATE goal weight.

4. MANEUVERING COMMANDS AND MOVEMENT

The active action set selected only specifies the magnitude of change for each of the maneuvering parameters (course, speed and depth). Other factors must be considered to determine the direction of change.

If the active goal is to avoid grounding:

Turn to the right to reverse course

Slow in speed

Reduce operating depth (drive toward the surface)

If the active goal is to avoid collision:

Turn to the right if obstacle is to the left of current COURSE

Turn to the left if obstacle is to the right of current COURSE

Slow in speed to minimize closure

Maintain operating depth

If the active goal is to remain within the operating area boundaries:

Turn to the right if boundary is to the left of current COURSE

Turn to the left if boundary is to the right of current COURSE

Slow in speed to minimize closure

Maintain operating depth

If the active goal is to transit to the destination:

Turn to the right if the destination point is to the right of current COURSE

Turn to the left if the destination point is to the left of current COURSE

Increase speed to minimize time to reach destination

Increase operating depth to meet secondary goal of operating as deep as possible

Now that both the magnitude and direction of maneuvering parameter changes have been determined, the submarine's new position can be calculated. The submarine is moved to that position, the sensing systems collect and provide new information to the OODDM, and the process repeats itself.

LIST OF REFERENCES

Archer, S., Warwick, W. & Oster, A., "Current Efforts to Model Human Decision Making in a Military Environment," *Proceedings of the Advanced Simulation Technologies Conference (ASTC)*, 2000.

Dreyfus H., *Intuitive, Deliberative, and Calculative Models of Expert Performance* in Zsambock, C. & Klein, G., *Naturalistic Decision Making*, Lawrence Erlbaum, 1997.

Eggleson, B. & Young, M., "Distributed Cognition (D-COG): A Cognitive Systems Engineering Based Approach to Human Work Modeling," *Proceedings of the XIVth Triennial Congress of the International Ergonomics Association and 44th Annual Meeting of the Human Factors and Ergonomics Society (IEA/HFES)*, 2000.

Ferber, J., *Multi-Agent Systems*, Addison Wesley Longman, 1999.

Forgy, C., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, 19 (1), 1982.

Haykin, S., *Neural Networks: A Comprehensive Foundation*, MacMillan, 1994.

Pew, R. & Mavor A., *Representing Human Behavior in Military Situations: Interim Report*, National Academy Press, 1997.

Pew, R. & Mavor A., *Modeling Human and Organizational Behavior*, National Academy Press, 1998.

Zsambock, C., *Naturalistic Decision Making: Where Are We Now?* in Zsambock, C. & Klein, G., *Naturalistic Decision Making*, Lawrence Erlbaum, 1997.

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

Deitel, H. & Deitel, P., *Java™ How To Program*, Prentice Hall, 1999.

Friedman-Hill, E., *Jess, The Expert System Shell for the Java Platform*, Sandia National Laboratories, 2001.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Dr. Michael Zyda
Naval Postgraduate School
Monterey, California
4. Dr. Rudolph Darken
Naval Postgraduate School
Monterey, California
5. Mr. Barry Peterson
Naval Postgraduate School
Monterey, California
6. LCDR Craig Oeltjen
Glenwood, Minnesota